AFRL-IF-RS-TR-2005-291
Final Technical Report
August 2005

SCALABLE ONLINE NETWORK MODELING AND SIMULATION

Rensselaer Polytechnic Institute

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K150

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-291 has been reviewed and is approved for publication

APPROVED: /s/

KEVIN A. KWIAT
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	AUGUST 2005	Final Jun 00 – Feb 05

4. TITLE AND SUBTITLE
SCALABLE ONLINE NETWORK MODELING AND SIMULATION

5. FUNDING NUMBERS
C - F30602-00-2-0537
PE - 62301E
PR - K150
TA - 01
WU - A1

6. AUTHOR(S)
Boleslaw Szymanski, Shivkumar Kalyanaraman,
Biplab Sikdar and Christopher Carothers

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rensselaer Polytechnical Institute
Department of Computer Science, Lally Hall
110 Eighth Street
Troy New York 12180-3590

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency AFRL/IFGA
3701 North Fairfax Drive 525 Brooks Road
Arlington Virginia 22203-1714 Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-291

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Kevin A. Kwiat/IFGA/(315) 330-1692/ Kevin.Kwiat@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

Performance analysis of large-scale protocol designs and network operations is accelerated by the novel, highly scalable and efficient parallel network simulators developed in this project. Also accomplished in this effort was: theoretical work on traffic modeling and BGP routing issues; and Quality of Service (QoS) as determined by stable assurances on loss rate, bandwidth and delay. The QoS problem addressed by this project had been previously unsolved for multi-domain IP-based inter-networks primarily because of complex deployment and coordination issues.

14. SUBJECT TERMS
Network, Modeling, Simulation, QoS

15. NUMBER OF PAGES
38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1. Summary

Performance analysis techniques are fundamental to aid the process of large-scale protocol design and network operations. If successful, such tools allow the system and network administrators to gain varying degrees of qualitative and quantitative understanding of the behavior of the system-under-test. A number of specific lower-level objectives include: validation of protocol design and performance for a wide range of parameter values (parameter sensitivity), understanding of protocol stability and dynamics, and studying feature interactions between protocols. Broadly, this is a quest for general invariant relationships between network parameters and protocol dynamics.

However, behavior of networking protocols is predicated on several parameters and has a bumpy "search space". Our project developed a solution to this dilemma: we view each simulation as a random experiment that gives some incremental information about system behavior and design the series of experiments to maximize the pace of learning about system behavior with the minimal number of experiments. To speed up the search and learning process from those experiments, we designed novel, highly scalable and efficient parallel network simulators. Those include the system for integration of simulations (even done with various simulators) and models using the Genesis system, as well as the advancement in the state of the art in the parallel simulation of network protocols under the optimistic protocol in the form of the ROSS system. Genesis imposes very low requirements on synchronization, communication and memory requirements of the executing hardware, yet supports superlinear speedup of the computations over the increasing number of processors. ROSS incorporates two novel techniques for optimistic parallel network simulations: light-weighted model implementation framework and “reverse computation” for rollback of the events when out of order event is received by a parallel processor.

There was also theoretical work done in traffic modeling and BGP routing issues. Finally, our work addressed the critical issue of Quality of Service (QoS) that was unsolved for multi-domain IP-based inter-networks primarily because of complex deployment and coordination issues. By “QoS” we refer to stable assurances on loss rate, bandwidth and delay.

2. Introduction

Performance analysis techniques are fundamental to aid the process of large-scale protocol design and network operations. There has been a tremendous explosion in the variety of tools and platforms available (e.g., ns-2, SSFNet, Click Router toolkit, Emulab, Planetlab). The high-level motivation behind the use of these tools is simple: to gain varying degrees of qualitative and quantitative understanding of the behavior of the system-under-test. A number of specific lower-level objectives include: validation of protocol design and performance for a wide range of parameter values (parameter sensitivity), understanding of protocol stability and dynamics, and studying feature interactions between protocols. Broadly, we may summarize the objective as a quest for general invariant relationships between network parameters and protocol dynamics.

However, the community still looks at the sample results obtained from such tools with skepticism because they are isolated (potentially random) and may not be representative of the real world. Behavior of networking protocols is predicated on several parameters and has a bumpy "search space" (see Figure 1). The need for scalable simulation and meta-simulation tools is implicit in the following statement: *"...The design of the Internet continues to evolve, and many aspects of its behavior are poorly understood… simulation plays a vital role in attempting to characterize how different facets of the network behave, and how proposed changes might affect the network's different properties. Yet simulating different aspects of the Internet is exceedingly difficult..."* (V. Paxon and S. Floyd, *Why We Don't Know How To Simulate The Internet,* Proc. WSC97).

Our project developed a solution to this dilemma: view each simulation as a random experiment that gives some incremental information about system behavior, and design the series of experiments to maximize the pace of learning about system behavior with the minimal number of experiments. This area of study is called "experiment design". Unfortunately, experiment design techniques of the past worked only on the small scale.

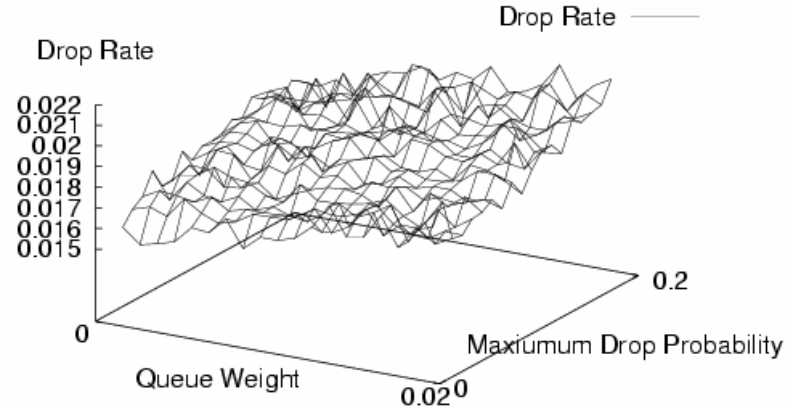

Figure 1. An Example of Network Protocol "Bumpy" Search Space as a Function of its Parameters.

In our project, we addressed the problem of large-scale experiment design with techniques that extract maximum information and confidence from a minimum number of carefully designed experiments. Such techniques can be used to find "good" results fast to guide either incremental protocol design or operational parameter tuning. One key result of this work is the development of a new algorithm, *Recursive Random Search*, a "Swiss army-knife" type heuristic optimization and experiment design algorithm that outperforms all comparable techniques (e.g., simulated annealing, genetic algorithms, table search, multi-start hill climbing). This part of the project was led by co-PI, Prof. Shivkumar Kalyanaraman.

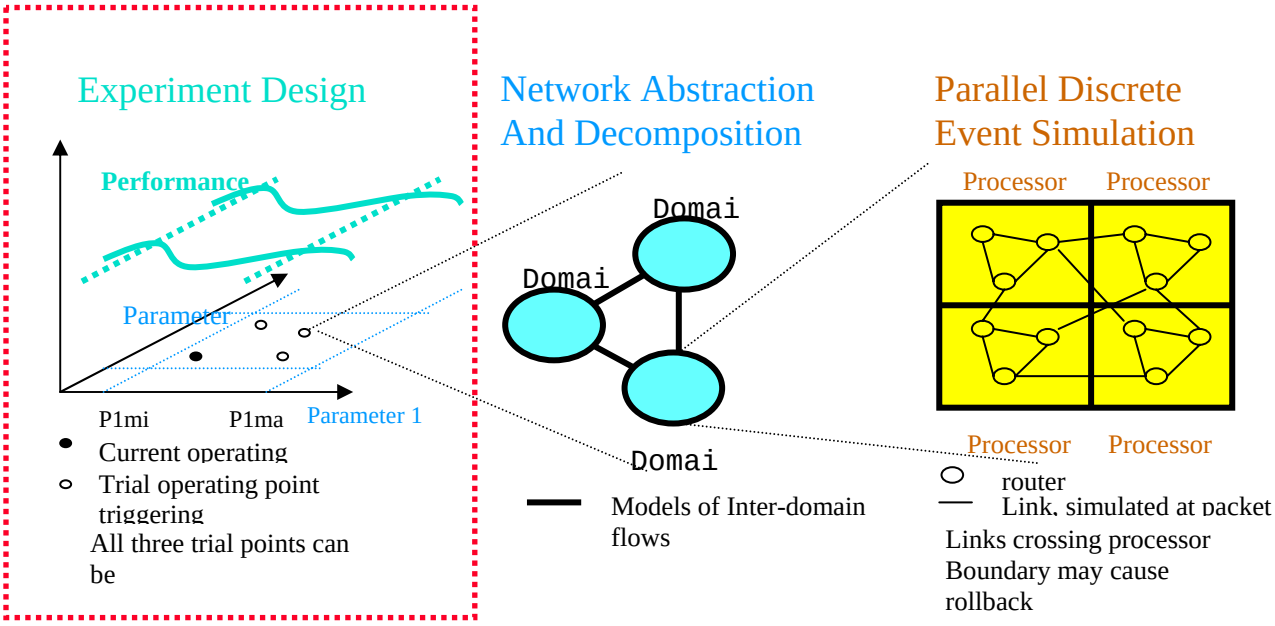

Figure 2. An Experiment Design as a "Meta-Simulation" Tool that Decides the Sequence of Simulation Experiments that Maximizes Learning about the Underlying Network.

As shown in the Figure 2, each "experiment" chosen by our framework (RRS or USF) may actually be a large-scale simulation! The issues of large-scale simulation have been addressed by other co-PIs in our project and included the development of a novel, optimistic parallel network simulator called ROSS.Net and design and implementation of the network decomposition and abstraction tool called Genesis. Genesis is capable of integrating different simulators and/or models in one large network simulation.

The way experiment design fits into the larger online simulation context is shown in Figure 3.

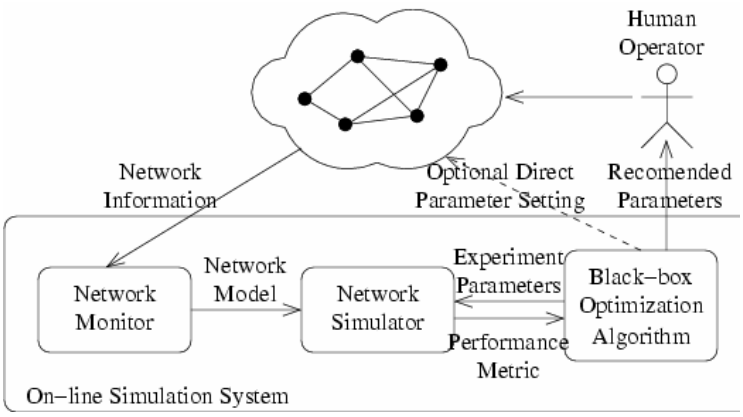

Figure 3. An illustration of How Online Simulation & Experiment Design (Black-Box Optimization) interfaces with a network.

In simulating large-scale networks at the packet level, one major difficulty is the enormous computational power needed to execute all events that packets undergo in the network. Conventional simulation techniques require tight synchronization for each individual event that crosses the processor boundary. The inherent characteristics of network simulations are the fine granularity of events (individual packet transitions in a network) and the high frequency of events that cross the boundaries of parallel simulations. These two factors severely limit parallel efficiency of the network simulation executed under the traditional protocols.

To satisfy the demand for scalable and efficient network simulation while avoiding the above mentioned pitfalls, we developed a novel approach to scalability and efficiency of parallel network simulation that partitions the networks into domains and simulation time into intervals. Each domain is simulated independently of and concurrently with the others over the same simulation time interval. At the end of each interval, traffic statistics data, including per flow average packet delays and packet drop rates, are exchanged between domain simulators. The simulators iterate over the same time interval until the exchanged information converges to the value within a prescribed precision before progress to the next time interval. This approach allows the parallelization with infrequent synchronization, and achieves significant simulation speedups. An overall execution scheme of this approach is presented in Figure 4.

Large memory sizes required by simulation software hinder the simulation of large-scale networks. To overcome this problem, our system supports distributed simulations in such a way that each participating simulator possesses only data related to the part of the network it simulates. This solution supports simulations of large-scale networks on machines with modest memory size. This work has been led by the PI of the entire project, Prof. Boleslaw Szymanski.

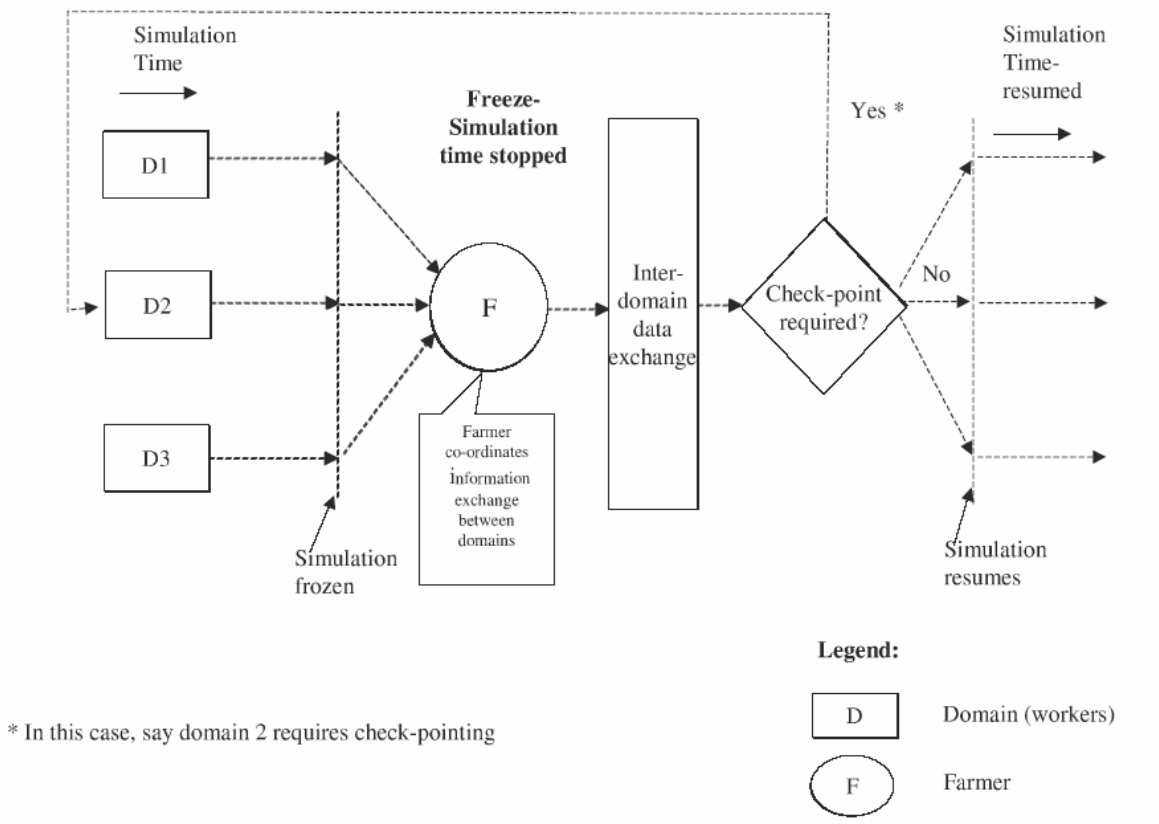

Figure 4. An Overall Execution Scheme in Genesis.

Another need addressed by our work includes the use of a large-scale parallel packet-level simulation of various networking protocols to understand their dynamics. For example, there are several issues in routing that need to be understood, such as cascading failures, inter/intra-domain routing stability, and interactions of policy-based routing with BGP features. One needs to perform large-scale simulations of inter-domain routing protocols along with various traffic engineering extensions, in order to observe their dynamics causing or effecting various performance problems in the current Internet.

We address this need using two techniques for parallel discrete event simulation of networks. First, we leverage an optimistic synchronization protocol to enable efficient execution on a hyper-threaded, multiprocessor system. Here, simulation objects, such as a host or router, are allowed to process events unsynchronized without regard for the underlying topology or timestamp distribution. If an out-of-order event computation is detected, the simulation object is rolled back and re-processed in the correct timestamp order using “reverse computation”. This approach greatly reduces the amount of state required to support optimistic event processing as well as increases performance. Second,

we devised an extremely light-weight model implementation framework called ROSSNet that is specifically designed for large-scale network simulation. If we examine state-of-the-art frameworks, such as ns, SSFNet, DaSSF, and PDNS, we find models that are highly detailed almost to the point of being full-protocol network emulators. For example, these frameworks provide support for a single end-host to have multiple interfaces, a full UNIX socket API for connecting to real applications, and other details that we believe are not necessarily relevant for large-scale simulation studies. The end result is that these systems require almost super-computer amounts of memory and processing power to execute large-scale models. In contrast, our framework implements minimum features needed to model network and be able to answer a particular protocol dynamics question in a large-scale scenario. This part of the project has been led by co-PI of this project, Prof. Christopher Carothers.

The theoretical part of our work focused on understanding and modeling the traffic characteristics in both wired and wireless networking environments as well as on network routing, and specifically on the dynamics of BGP, its analytic characterization and on developing techniques to improve its stability and convergence time. This part of work was led initially by the original co-PI of this project, Prof. Kenneth Vastola, replaced later by co-PI, Prof. Biplab Sikdar.

With significant progress in silicon technologies, storage media and signal processing in recent years, many applications, such as multimedia communication, become possible. Compared with traditional network file sharing or exchanging, multimedia communication is far more difficult because of its stringent requirements. For instance, Internet video streaming requires network to provide bandwidth, packet loss, delay and delay jitter guarantees, at least in a statistical sense. But today's Internet does not provide any of these guarantees. Researchers have tried to develop enabling technologies for bandwidth and delay guaranteed applications. But all of them have encountered their respective difficulties, either in algorithm complexity or in real deployment (especially in an inter-domain context).

In summary, the problem of Quality of Service (QoS) was unsolved for multi-domain IP-based inter-networks primarily due to complex deployment and coordination issues. By “QoS” we refer to stable assurances on loss rate, bandwidth and delay. In contrast, QoS within a single administrative network today is largely a non-issue due to declining bandwidth costs and availability of administrative controls. This part of the project addressed the critical issue of Quality of Service (QoS) in multi-domain IP-based inter-networks primarily and was led by co-PI, Prof. Shivkumar Kalyanaraman.

3. Methods, Assumptions, and Procedures

3.1 Recursive Random Search: Large-Scale Experiment Design for Online Simulation

Meta-simulation capabilities go beyond mere scaling of simulation platforms and providing large-scale experiment design. Statistical experiment design considers the system-under-test as a black-box that *transforms input parameters to output metrics*. The goal of experiment design is to *maximally characterize* the black-box with the *minimum number of experiments*. Another goal is *robust* characterization, i.e., one that is minimally affected by external sources of variability and uncontrollable parameters, and can be specified at a level of confidence. Beyond characterization, the methodology aims to *optimize* the system, i.e. allows one to find the appropriate input parameter vector that elicits the best output response. The underlying premise of experiment design is that each experiment (e.g., a simulation run, an Emulab or Planetlab test run) has a non-negligible cost.

Parameter configuration is a common procedure used in large-scale network protocols to support multiple operational goals. This problem can be formulated as a black-box optimization problem and solved with an efficient search algorithm. The method of choosing experiments (or simulations) to solve this problem is called "***experiment design***" (Figure 5). Experiment design is ***truly the "brains" behind a self-aware, self-tuning network***. In essence, experiment design chooses what simulations to run so that a high-level network design or configuration objective can be met. This is important for large-scale parameter configuration, and involves the global search in a very-large-dimensional state space.

In this project we have developed a heuristic Recursive Random Search (RRS), for large-scale network parameter optimization (see Figure 5). RRS is meant for global optimization problems with *little or no information known about the problem structure*. Therefore it has wide applicability. RRS is also particularly useful for large-dimensional problems (e.g., 1000s-millions of parameter dimensions) and has the property of being able to *find "good" solutions "fast"*, i.e., it displays a high degree of *efficiency* compared to known search techniques (e.g., multi-start hill climbing, genetic algorithms, simulated annealing, tabu search, etc).

The RRS algorithm is based on the initial high-efficiency property of random sampling and attempts to maintain this high-efficiency by constantly "restarting" random sampling with adjusted sample spaces. Due to its root in random sampling, the RRS algorithm is robust to the effect of random noises in the objective function and is advantageous in optimizing the objective function with negligible parameters. These features are very important for the efficient parameter optimization of network protocols. The performance of RRS has been demonstrated with the tests on a suite of benchmark functions.

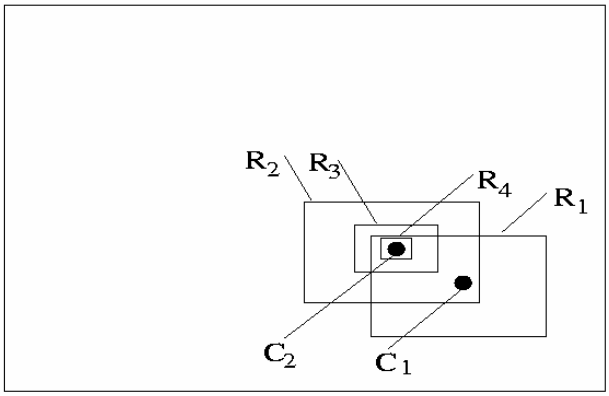

Figure 5. Schematic Illustrating the Recursive Random Search (RRS) Technique: Re-scaling the Search Space and Using Random Sampling Throughout.

We have conducted scaling tests with RRS to handle *thousands* of parameter dimensions in *very difficult* test optimization benchmarks (see example in Figure 6). The RRS algorithm has been applied to the adaptive configuration of several real-world network protocols, such as RED, OSPF and BGP. The particular problems include RED parameter tuning, OSPF link weight tuning for traffic engineering, and BGP outbound load balancing using LOCAL_PREF tuning.

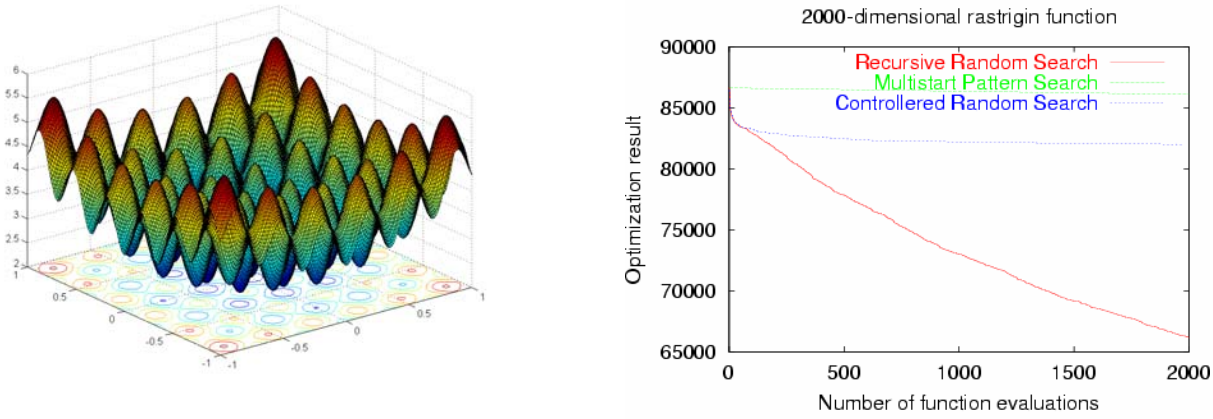

Figure 6. Complex Benchmarks and Large-Dimensional Test Results with RRS.

An application to RRS to a real network online simulation problem is illustrated in Figure 7. In the middle of each simulation, the RRS system kicks in to improve performance.

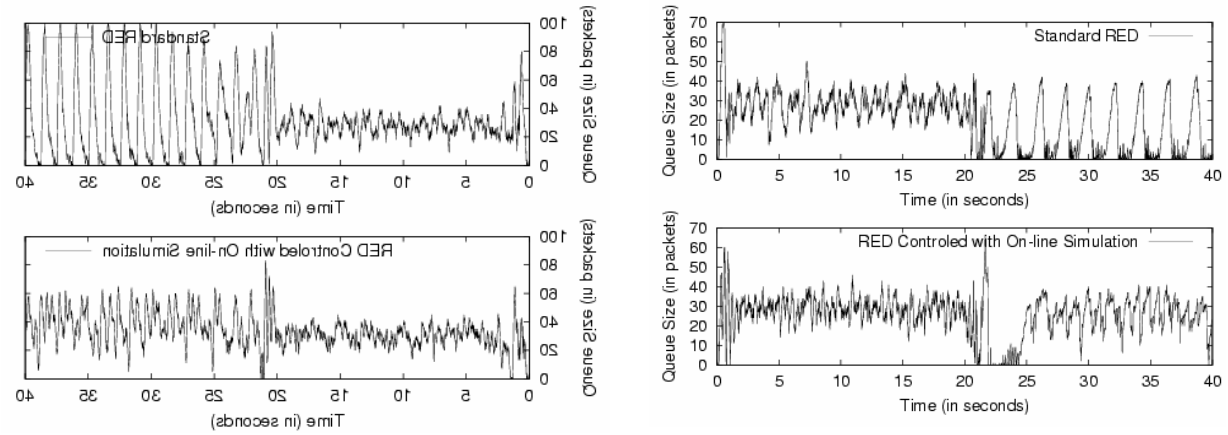

Figure 7. An Application of RRS to a Real Network to Improve Performance.

Our recursive random search (RRS) algorithm has been extended to make it more general. This approach is called *Unified Search Framework (USF).* Unified Search Framework (USF) is a general, component-based, parallel optimization platform (see Figure 8). Currently, when given a specific optimization problem, one has to choose or design one algorithm, then implement it to apply to the problem. USF, in contrast, can be used to easily construct ***on-the-fly***, a tailored optimization algorithm for a specific problem, that is run in a parallel way to fully utilize available computing resources. We have tested the USF on some benchmark functions and got very promising test results.

The main advantage of the USF is that it can function as a general platform to address various practical optimization problems with different features. It is especially suitable for those large-scale problems with simulation for function evaluations, such as the ones in on-line network parameter optimization. In future work, we will apply the USF to a variety of large-scale network optimization problems and demonstrate the flexibility of the USF in handling different problems.

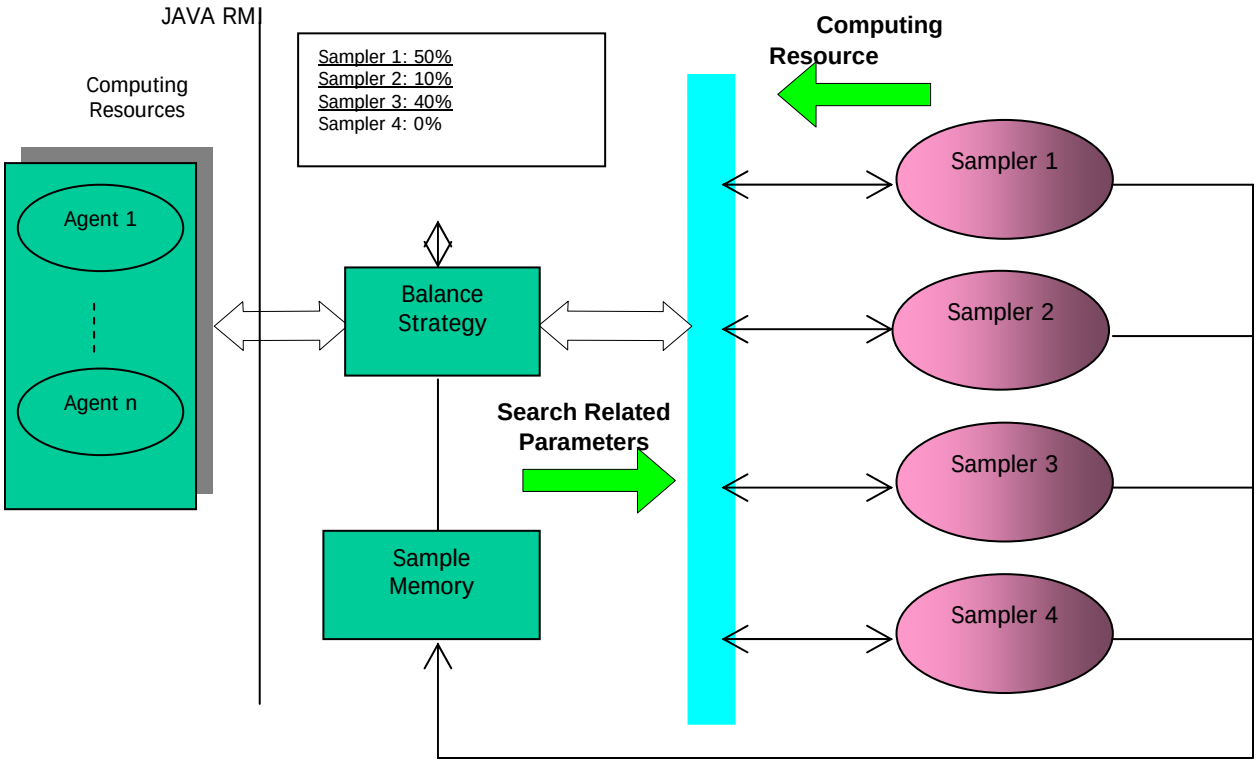

Figure 8. Schematic of the Unified Search Framework (USF).

3.2. Genesis

In simulating large-scale networks at the packet level, one major difficulty is the enormous computational power needed to execute all events that packets undergo in the network. Conventional simulation techniques require tight synchronization for each individual event that crosses the processor boundary. The inherent characteristics of network simulations are the fine granularity of events (individual packet transitions in a network) and high frequency of events that cross the boundaries of parallel simulations. These two factors severely limit parallel efficiency of the network simulation executed under the traditional protocols.

Another difficulty is the large memory size required by large-scale network simulations. With the current trend of simulating ever larger and more complicated networks, the memory size becomes a bottleneck. Centralized network configuration and routing information results in large memory requirements during construction of the simulated network. Additionally, the needed memory increases also with the intensity of traffic flows that dictate the size of the future event list. We believe that to simulate truly large networks, the comprehensive, distributed memory approach is needed.

This part of a project focused on developing an architecture that can efficiently simulate very large heterogeneous networks in near real time. Our approach combines simulation and modeling in a single execution. The network parts, called domains, are simulated at the packet level concurrently with each other. Each domain maintains the model of the network external to itself that is built by using periodically output from other domain simulations. If this model is faithfully representing the network, the domain simulation will exactly represent the behavior of its domain, so its output will support the correct models maintained by other simulations. Each domain simulation repeats its execution, each time adjusting its model of the rest of the network and feeding the other simulations with increasingly precise model of its own domain. As a result, all domain simulations collectively converge to the consistent state of all domains and all models.

Thanks to the coarse granularity synchronization mechanism used in this system, it is able to use different simulators in a single, coherent network simulation, hence we called it General Network Simulation Integration System, or *Genesis* in short.

Genesis also addresses the large memory requirement problem in large-scale network simulations. Many parallel simulation systems are capable of speeding-up simulation time. However, to accomplish such a speedup, they require that every machine involved has a big enough memory to hold the full network. This requirement is most easily achieved through a system with shared memory. In Genesis, in contrast, memory usage is fully distributed. Each Genesis domain simulator stores only a part of the network, together with some additional information that is required to cooperate with other simulators. In such a way, large networks can be simulated by clusters of machines, in which each machine has only a small memory.

3.2.1 Genesis Approach

Because of the huge amount of events and high event rate involved in large-scale network simulations, parallel and distributed simulation techniques introduce high synchronization overhead. This overhead comes from a “general rule” for parallel and distributed simulations: each event that is created on one processor and needs to be executed on the other introduces synchronization overhead. The processors involved in such an event need to be synchronized for this event and this delays their execution. This general rule limits the improvement of synchronization performance for network simulation. Can we break this “general rule”? Our efforts to find the answer for this question had led us to the research presented here.

In the traditional view of network simulation, we consider a group of parallel or distributed simulation sub-systems as one simulation system that is required to produce exactly the same simulation result as a sequential simulation. However, in many network simulation applications, we do not care what may have happened to individual network packets. Instead, we are more interested in some “metrics”, for example, traffic throughput, end-to-end packet delay, packet lost rate, et cetera. Thus, from the higher-

level view, we are running simulations to achieve statistics data for the “metrics” we are interested in. A simulation system only needs to produce these data accurately, or with approximations within a satisfactory range, instead of guaranteeing the correct behavior of each individual packet. This gave us the possibility to simplify a network simulation.

With this novel view of network simulation, we consider a distributed simulation system as a loosely coupled distributed computing system. Each distributed domain simulator runs separately doing local computation (simulating the domain assigned to it) within a period of time, with all the information of the network it has at that time, to produce local results as accurately as possible. Periodically, these distributed simulator exchanges computation results and updates network information among them. Each simulator uses “fresh” information to update its own computation and information base, to produce more accurate results during the next iteration. In this way, we don't need to synchronize and exchange data among simulators at event-level (packet-level). The synchronization for these loosely coupled sub-systems can be very infrequent and the overhead can be reduced significantly.

In Genesis, a large network is decomposed into parts and each part is simulated independently and simultaneously with the others. Each part represents a subnet or a sub-domain of the entire network. These parts are connected to each other through edges that represent communication links existing in the simulated network. In addition, we partition the total simulation time into separate simulation time intervals selected in such a way that the traffic characteristics change slowly during most of the time intervals.

Each domain is simulated by a separate simulator that has a full description of the flows whose sources are within that domain. This simulator also needs to simulate and estimate flows whose sources are external to the domain but will be routed to or through the domain. In addition to the nodes that belong to the domain by the user designation, we also create *domain closure* that includes all the sources of flows that reach or pass through this domain. Since these are copies of nodes active in other domains, we call them *proxy sources*. Each proxy source uses the flow definition from the simulation configuration file.

The flow delay and the packet drop rate experienced by the flows outside the domain are simulated by the random delay and probabilistic loss applied to each packet traversing the in-link proxy. These values are generated according to the average packet delay as well as observed packet loss frequency communicated to the simulator by its peers at the end of simulation of each time interval. Each simulator collects this data for all of its own out-link proxies when packets reach the destination proxy.

A Farmer-Worker system is designed for data exchange among these domain simulators. Each domain simulator runs as a worker, and one stand-alone server runs as a farmer to synchronize domain simulators. Every domain simulator stops its simulation at pre-defined checkpoints, and exchanges data with all the other domain simulators. During a

checkpoint, each domain simulator also checks its convergence condition by analyzing the received data, based on some pre-defined metrics (end-to-end packet delay, packet loss rate, etc.) and parameters (e.g. precision threshold). The farmer collects convergence information from all domain simulators and makes global convergence decisions. If some convergence condition is not satisfied, the farmer will inform some or all domain simulators to roll back and re-iterate. Those simulators that need to roll back will go back to the last checkpoint and re-simulate the last time interval, however, utilizing the data received during the current checkpoint. When all the domain simulators converge, a global convergence is reached and the farmer will inform all the domain simulators to go on to the next time interval. The system framework is shown in Figure 4.

Genesis uses a coarse granularity synchronization mechanism, described above, to simulate network traffics, e.g., TCP or UDP flows. This is achieved by having parallel simulators loosely cooperating with each other in the Farmer-Worker framework. They simulate partitioned network concurrently with and independently of each other in each iteration. They exchange data only during the checkpoints executed between iterations. In addition, individual packets are not stored or exchanged among parallel simulators. Instead, each data flow is summarized based on some pre-defined metrics, and only the summarized traffic information is exchanged among parallel simulators.

This approach avoids frequent synchronization of parallel simulators. Parallel domain simulators are running independently. Each of them uses data that it received from others to represent the external network outside of its own domain. By periodically exchanging data with other domain simulators and reiterating over the same simulation time interval to achieve a global convergence, the simulation of the whole network approximates the sequential simulation of the same network with controllable precision.

We believe that communication networks simulated this way will converge thanks to the fact that path delays and packet drop probabilities are monotonic functions of the traffic intensity (congestion). For example, if in iteration I_k a part N_i of the network receives more packets than the sequential simulation would deliver, then this part will produce fewer packets than the sequential simulation would. These packets will create inflows in the iteration I_{k+1}. Clearly then, the sequential simulation will deliver the number of packets that is bounded from above and below by the numbers of packets generated in two subsequent iterations I_k and I_{k+1}. Hence, in general, iterations will produce alternately too few and too many packets in the inflows providing the bounds for the number of packets in the sequential simulation. By selecting the middle of each pair of bounds, the number of steps needed to convergence can be limited to the order of logarithm of the needed accuracy, so convergence is quite fast. In our experiments, the convergence for UDP traffic was achieved in 2 to 3 iterations, for TCP or mixed UDP/TCP traffic in 5-10 iterations, and for BGP/TCP/UDP traffic it was about twice the number of Autonomous Systems simulated.

One issue of great importance for efficiency of the described method is frequency of synchronization between simulators of parts of the decomposed network. Shorter synchronization time limits parallelism but decreases also the number of iterations necessary for convergence to the solution because changes to the path delays are smaller. Variance of the path delay of each flow can be used to adaptively define the time of the synchronization for the subsequent iteration or the simulation step.

It is easy to observe that the execution time of a network simulation grows faster than linearly with the size of the network. Theoretical analysis supports this observation because for the network size of order $O(n)$, the sequential simulation time include terms that are of order $O(1)$ to $O(n*log(n))$, that correspond to processing events in the order of their simulation time in the event queue (depending on queue types); $O(n(log(n))^2)$ to $O(n^2)$ (depending on the model of the network growth) that result from number and complexity of events that packets undergo flowing from source to destination. The average length of a path traversed by each packet, the number of active flow sources, the number of flows generated by each source and even the number of packets in each flow may grow at the rate of $O(log(n))$ to $O(n^a)$, where $0.5<a<1$, as the function of n, the number of nodes in the network. They together create the super-linear growth in the number of the events processed by the simulation. From the above, we can conclude that the execution time of a network simulation is a superlinear function of the network size. Therefore, it is possible to speed up the network simulation more than linearly by splitting a large simulation into smaller pieces and parallelizing the execution of these pieces.

Another advantage of the proposed method is that it is independent of any specific simulator technique employed to run simulators of the parts of the decomposed network. Rather, it is a scheme for efficient parallelization based on convergence to the fixed-point solution of inter-part traffic. The convergence is measured by a set of parameters characterizing the traffic rather than individual packets. Our primary application is network management based on on-line network monitoring and on-line simulation. The presented method fits very well to such an application as it predicts changes in the network performance caused by tuning of the network parameters. Hence, the fixed-point solution found by our method is, with high probability the point into which the real network will evolve. However, there are open questions such as under what conditions the fixed-point solution is unique, or when the solution found by the fixed-point method coincide with the operating point of the real network.

In many network simulation scenarios, the real data of the traffics packets are not important to the simulation result, so no individual packet is synchronized between two parallel simulations in UDP and TCP traffic simulations. Instead, packets are "summarized" on some metrics (delay, drop rate, etc.). Only these data are exchanged between domains at the end of each time interval. This approach was designed to simulate TCP and UDP data traffics, but could not be used to simulate some other flows, for example, data flows providing information for routing protocols. This is because the

traffic of a routing protocol cannot be summarized; instead, different content and timing of each routing packet might change the network status. Particularly, our desire to simulate the BGP protocol required us to develop additional synchronization mechanism in Genesis.

We developed an event-level synchronization mechanism that can work within the framework of Genesis and support the simulation of the BGP protocol. To simulate a network running the BGP protocol for inter-AS (Autonomous System) routing, with background TCP or UDP traffics, we decompose the network along the boundaries of AS domains. Each parallel simulator simulates one AS domain, and loosely cooperates with other simulators. When there are BGP update messages that need to be delivered to neighbor AS domains, the new synchronization mechanism in Genesis guarantees that these messages will be delivered in the correct time-stamp order. At the synchronization point in Genesis, the BGP update messages collected in the proxy BGP routers, if there are any, are forwarded to the corresponding destination AS domain simulators through a component called BGP agent. These update messages are delivered to the BGP agent in the destination AS domain through a Farmer-Agent framework, and are distributed there to the BGP routers which are the destinations of these messages.

Simulations of large-scale networks require large memory sizes. This requirement can become a bottleneck of scalability when the size or the complexity of the network increases. For example, ns2 uses centralized memory during simulation, which makes it susceptible to the memory size limitation. We believe that the only permanent solution to the simulation memory bottleneck is to develop the distributed memory approach.

In a typical parallel network simulation using non-distributed memory, each of the parallel simulators has to construct the full network and to store all dynamic information (e.g., routing information) for the whole network during the simulation. To avoid such replication of memory, we developed an approach that completely distributes network information. Thanks to this solution, Genesis is able to simulate large networks using a cluster of computers with smaller dedicated memory (compared to the memory size required by shared memory-based SSFNet simulating the same network).

To test the performance of our solution, we run BGP network simulations in which the first round of BGP update message burst happens when AS domains start to exchange BGP information to set up the global inter-AS routing. In Genesis, AS domains are simulated distributively, and BGP update messages are synchronized by re-iterating over one time interval until the BGP messages exchanged among domains converge (no more changes) on that interval. When a BGP update message propagation happens, Genesis re-iterates over the same time interval until the propagation converges. Each re-iteration consumes simulation run time. When BGP update message propagations happen periodically during the simulation, the additional run time required by these re-iterations will decrease the speedups achieved by utilizing parallel simulation. An interesting question is how such on-going BGP activities would affect the simulation performance.

To investigate this question, we introduced BGP session crashes into our experiments. The simulation time was fixed at 400 seconds for 20 AS's and, correspondingly 20 Sun 10 Ultrasparc workstations. The BGP session between two neighboring AS domains, campus network 3 and 4, crashed every *D* seconds, each time staying down for *D/2* seconds, and then coming back and staying alive for another *D/2* seconds.

As expected, in the simulation time intervals in which the specified BGP session went down, BGP update messages were propagated causing the broken routes to be withdrawn and back up routes being set up. Accordingly, data packet flows also changed and used the new routes. When that BGP session came back again, BGP update messages propagated again and re-established the broken routes. In either case, the relevant time interval had to be re-iterated again and again until it converged. So these intervals were "slow-down" periods. The time intervals with no BGP propagations were "speed-up" periods thanks to the novel parallel simulation mechanism used by Genesis. As a result, the proportion of the "slow-down" time in the whole simulation time affects the overall speedup: the less frequently the crashes happen, the greater the speedup that can be achieved. When the frequency of BGP crashes is not high, these "slow-down" periods will not significantly slow down the progress of the simulation.

On the other hand, the length of the time interval also affects the total re-iteration time. Ideally, smaller the length of the time interval, shorter the total re-iteration time. But there are two other factors which benefit from longer intervals. First, small interval length increases the synchronization and checkpointing overheads between intervals and can overwhelm the simulation speedup. Second, if the interval length is too small to cover the BGP propagation period, then the next time interval will need to be re-iterated in addition to the current one. We varied the value of *D* from 80 to 60, 40 and then 20 seconds, and also used different lengths of iteration time intervals for Genesis checkpointing, denoted as *T*, which was set to 20, 10 and 5 seconds. Figure 9 shows the speedups achieved in these experiments.

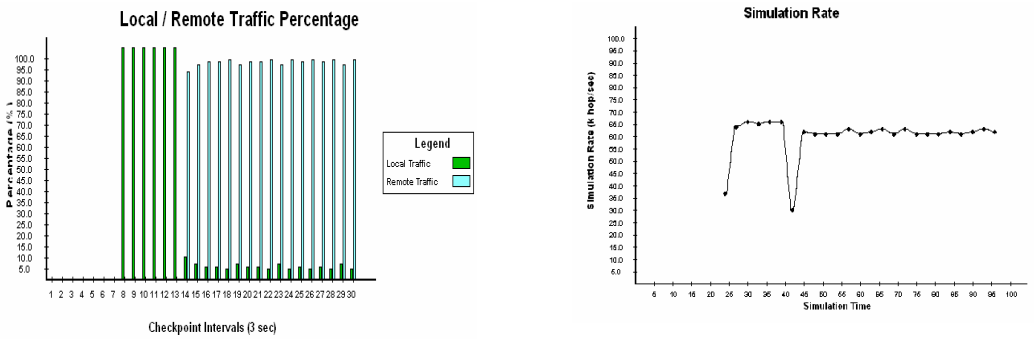

Figure 9. Distributed UDP Flooding Simulation with U.S. Backbone Net Topology.

Java-based SSFNet and C++/TCL-based ns2 use different network models and different simulation frameworks. The details of the implementation of traffic packets and other network entities are different in these two systems. Thanks to the coarse granularity synchronization framework in Genesis, only traffic statistics data summarized on some metrics are exchanged among domain simulators, while the implementation details of the actual network traffic in one domain can be viewed as a black box to the other. This facilitates the design of a general integration framework.

In Genesis, we designed the general format of the traffic statistics data message being exchanged in the framework, and the general conventions for a domain simulator to identify a network entity (e.g., nodes identified by a global node id). Then, the rest of the work is the implementation of conversion between native data format and the general message format for both SSFNet and ns2. Because of this general inter-operation interface an SSFNet domain simulator can work with either SSFNet or ns2 domain simulators in exactly the same way. Another advantage of this approach is its extensibility: any domain simulators complying with this general interface can be easily plugged into Genesis.

Based on the design of interoperability between ns2 and SSFNet, we adopted a similar approach to enable interoperability between SSFNet and GloMoSim. We created a scenario where we have mixed-mode traffic between a wired network (modeled using SSFNet) and a wireless network (modeled using GloMoSim). The SSFNet part of the network views the wireless GloMoSim domains as a single node proxy network, which is the source and sink for all traffic originating and destined respectively to the latter. Similarly, for GloMoSim, the SSFNet domains are represented by a single node proxy network as well. At each checkpoint interval, the information about inter-domain traffic statistics data is exchanged between SSFNet and GloMoSim simulations. The receiving simulation uses this information to adjust the single node proxy network and the links connecting to it, to better represent its cooperating simulation. And based on the received information and local conditions in the domain, a decision whether to roll back or not is made by each of the domains.

3.2.2 Genesis Design Overview

Genesis took some common approaches for parallel and distributed simulation systems and had all the general components for these systems, although it adjusted them to meet the special needs of coarse granularity synchronization. In conventional parallel or distributed simulation that uses the space partitioning technique to divide network into domains, the system usually consists of these general components:

- Network partitioning. The network topology being simulated is logically partitioned into areas, and each area is assigned to one processor. The simulation script that defines the network provides some functionalities to divide the network and assign processors.

- Concurrent Simulation. Network areas are simulated concurrently on different processors. Each processor simulates only the part of the network assigned to it and handles events generated from this part of network, or events received from other processors.
- Data management. In conventional simulation, the simulation data exchanged among processors are events. Events originated from one processor and targeted to another processor are remote events. The parallel or distributed simulation system should recognize these remote events and forward them to the correct destination, by using either shared memory or explicit information exchanging techniques (e.g. MPI, socket connection).
- Time management. Parallel or distributed simulators need to be synchronized. As we explained earlier, different synchronization approaches are designed to achieve the same goal that in each processor, events are handled in the correct order of their time-stamps.

In our novel simulation system using coarse granularity synchronization technique, there are differences in the roles and functions of these components:

- Network partitioning. A network topology is partitioned in the same way as in conventional simulations, and each network partition is assigned to one processor. However, it does not mean that one processor will only simulate the partition assigned to it. Instead, the assigned partition is the "simulation focus" of this processor, and fragments of other partitions related to this one will also be simulated in this processor.
- Concurrent Simulation. Each processor simulates the network partition assigned to it in detail the same way as conventional simulation systems. However, processors do not exchange remote events among each other. Instead, each processor contains not only its own part of the network, but also a simplified model of the rest of the network. Thus, a "remote" event related to the rest of the network can be delivered to the corresponding simplified network model. In this way, there is no need to exchange "remote events" all events are internal events to a processor.

- Data management. Data management in Genesis is different from conventional systems. No remote events need to be exchanged among processors. As explained above, for one processor, the simplified network model serves as a representation of the part of network simulated in details by other processors, in other words, the "outside world". In order to correctly represent the "outside world", each processor collects simulation statistics data from the part of network assigned to it and exchange them with other processors. Each processor then uses the data received from other processors to adjust the network model representing the "outside world".

- Time partitioning and management. Time management in Genesis is different because no remote events need to be synchronized. Instead, the simulation time is partition into intervals separated by checkpoints. During each checkpoint, the simulation time of every processor is synchronized and a convergence decision is made. Based on the received data from its peer domains, a domain simulator might need to re-iterate one simulation time interval to produce more accurate results.

In Genesis, network partitions are called *domains*. For one processor, the domain assigned to it is called an *active domain*, and the domains assigned to other processors are called *non-active domains*. In the active domain, the network structure is the same as the non-decomposed network. In non-active domains, traffic sources and destinations are represented by *proxy sources*, which can be activated or deactivated dynamically during the simulation. *Path shortcuts* are used to simplify any traffic paths in non-active domains. They are implemented as *proxy links* that connect *proxy sources* directly to border routers in the domain. Simulation statistics data are used to adjust these *proxy links* to represent network traffic changes during the simulation. Each processor is only responsible for the simulation of its active domain, and collects simulation statistics data within the active domain. However, the proxy source and proxy link structure for non-active domains is also essential that it completes the traffic path from the source node to the destination node. As the result, in Genesis, each processor simulates full traffic paths from source nodes to destination nodes. This is important because for TCP traffic source nodes must receive ACK packets from destination nodes to continue its packet sending.

The Genesis system model explained above introduced a new approach of constructing a distributed simulation system. However, Genesis is not only one network simulation system. Instead, it is a general approach that can be used to transform conventional sequential or parallel simulation systems into scalable distributed simulation systems, as well as constructing new systems from scratch. Besides this, different conventional systems transformed by the Genesis approach will be able to cooperate with each other.

The approach in Genesis described above was originally designed for protocols that generate packets without feedback flow control, such as Constant Bit Rate (CBR) UDP traffic. However, modeling the inter-domain traffic that uses feedback based flow control, such as any of many variants of TCP, requires more processing capabilities. To control congestion in a network or the Internet, some protocols use congestion feedback. The most important among them is the TCP protocol used in TCP/IP-based Internet congestion control. For our method, the important property of TCP traffic is that the rate of the source is dependent on the conditions not only in the source domain but also in all intermediate and destination domains of the traffic. Additionally, each data flow has a corresponding acknowledgment flow that paces the source. As a result, for the TCP traffic, the precision of our flow simulation depends on the quality of the replication of the round trip traffic by the packets and their acknowledgments. Moreover, the feedback loop for iterations is extended. For example, in two-domain TCP traffic, a change in

congestion in the source domain will impact delays of data packets in the destination domain in the following iteration and the delays of the acknowledgment packets in yet subsequent iteration. As a result, convergence is slower in simulation of networks with TCP flows. Still, a superlinear speedup has been achieved for a sample network of 64 nodes and large number of TCP and UDP flows (see Figure 10).

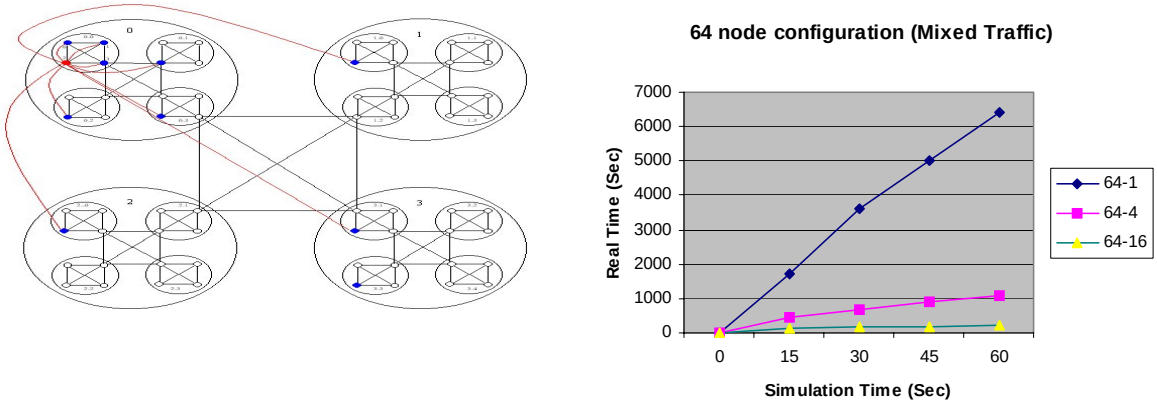

Figure 10. Superlinear Speedup for TCP and UDP Traffic in 64-node Network.

3.3. ROSS-Net

ROSS-Net is a novel parallel network simulation system using a discrete event simulation technique and optimistic protocol. It uses two new techniques that are essential to its efficiency and usability. First, ROSS-Net leverages an optimistic synchronization protocol to enable efficient execution on a hyper-threaded, multiprocessor system. In ROSS-Net, simulation objects, such as a host or router, are allowed to process events unsynchronized without regard for the underlying topology or timestamp distribution. If an out-of-order event computation is detected, the simulation object is rolled back and re-processed in the correct timestamp order. Unlike previous optimistic protocols, such as Time Warp, the rollback mechanism is realized using *reverse computation*. Here, events are literally allowed to execute backward to undo the computation. This approach greatly reduces the amount of state required to support optimistic event processing as well as increases performance.

Next, we devised an extremely light-weighted model implementation framework that is specifically designed for large-scale network simulation. If we examine state-of-the-art frameworks, such as Ns, SSFNet, DaSSF, and PDNS, we find models that are highly detailed almost to the point of being full-protocol network emulators. For example, these frameworks provide support for a single end-host to have multiple interfaces, a full UNIX socket API for connecting to real applications, and other details that we believe are not necessarily relevant for large-scale simulation studies. The end result is that these

systems require almost super-computer amounts of memory and processing power to execute large-scale models.

In contrast, our framework poses the question: “what does the user really need to model in order to answer a particular protocol dynamics question in a large-scale scenario?” For example, are all layers in a protocol stack really necessary? Can a host just be a TCP sender or just a TCP receiver? Does the simulated host really need to be both? By asking these kinds of questions, our framework enables a single TCP connection to be realized in just 320 bytes total (both sender and receiver) and 64 bytes per each packet-event.

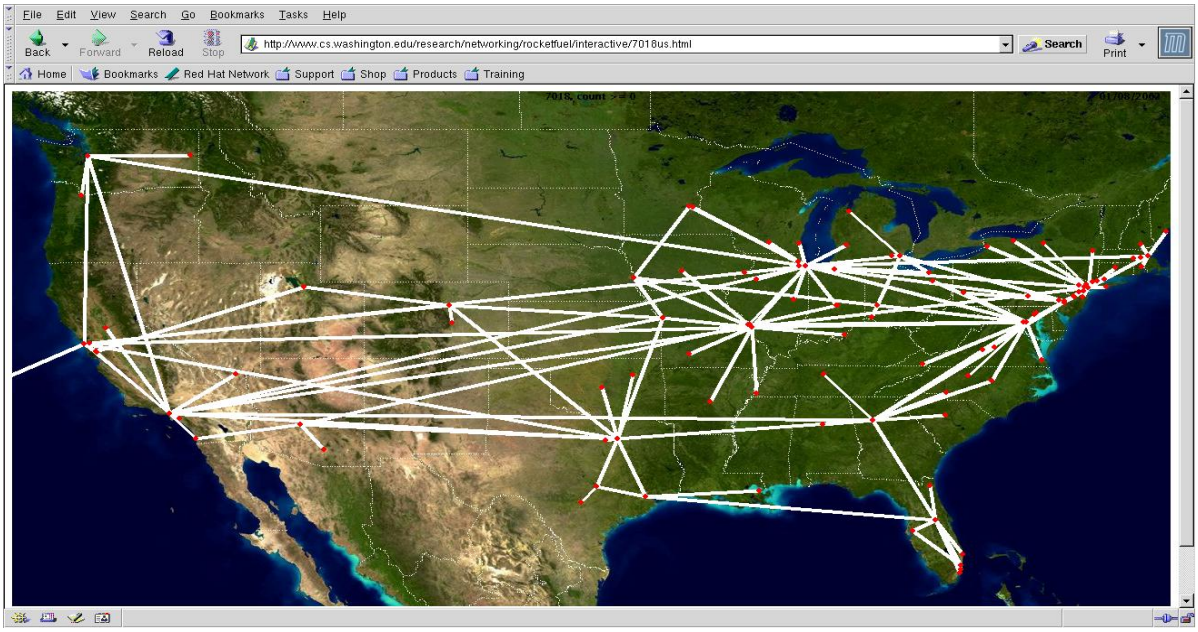

Figure 11. Topology of the AT&T US Network Simulated in Ross.Net.

These innovations enable the simulation of million-node network topologies using inexpensive commercial off-the-shelf multiprocessor systems consuming less than 1.4 GB of RAM in total. We have recently simulated the AT&T and several other large networks with full OSPF and TCP dynamics in a large-scale simulation platform. This configuration has tens of thousands of routers and over a million TCP flows interacting dynamically. This example suggests that we can now begin to simulate and make exploratory analysis of large-scale protocol feature interactions with our simulation and experimental design platform. Such complex feature interactions cannot be simulated, visualized or explored in small-scale ns-2 simulations.

3.4. Traffic and BGP Modeling

In traffic modeling, our work started in the wired networking domain where we investigated the causes of second order scaling in network traffic and proposed solutions for reducing the degree of scaling. We first showed, using analysis, measurements from real world traces as well as ns-2 simulations that timeouts and exponential backoffs are two of the main causes behind scaling a TCP traffic. We also showed that the degree of scaling is related to the loss rates seen by the TCP flows and the congestion on the links. These observations were then used to develop techniques, both at the user as well as network level, to reduce the degree of scaling a TCP traffic. We proposed and showed the effectiveness of TCP pacing and active queue management strategies to reduce second-order scaling a TCP traffic. In the next phase of our work, we developed analytic models for network traffic (more specifically, for the packet inter-arrival times) in IEEE 802.11 based wireless networks. We also showed the presence of scaling behavior in the traffic of such networks. The packet interarrival model was then extended to obtain the MAC layer packet service times and consequently the distribution function of the delay in 802.11 based networks (using discrete time G/G/1 queues).

In BGP routing, our initial work focused on simulations to obtain empirical relationships between the convergence times of a network of BGP routers and the stochastic properties of the random graph characterizing the network. We then developed analytic models to characterize the probability of the convergence times exceeding certain bounds based on the network properties. These models characterize the probability of successive expirations of the Minimum Route Advertisement Interval (MRAI) timer as a function of network topology (we use random graphs to characterize the network topologies). We have also developed techniques to reduce the convergence time of a network of BGP routers. These techniques are based on identifying the causes of redundant MRAI timer instantiations by BGP routers and then preventing them. Our results show considerable reduction in the convergence times when our proposed modifications are incorporated into BGP.

3.5 Overlay QoS Using Closed-Loop Control

With significant progress in silicon technologies, storage media and signal processing in recent years, many applications, such as multimedia communication, become possible. Compared to traditional network file sharing or exchanging, multimedia communication is far more difficult because of its stringent requirements. For instance, Internet video streaming requires network to provide bandwidth, packet loss, delay and delay jitter guarantees, at least in a statistical sense. But today's Internet does not provide any of these guarantees. Researchers have tried to develop enabling technologies for bandwidth and delay guaranteed applications, but all of the proposed techniques have encountered

their respective difficulties, either in algorithm complexity or in real deployment (especially in an inter-domain context).

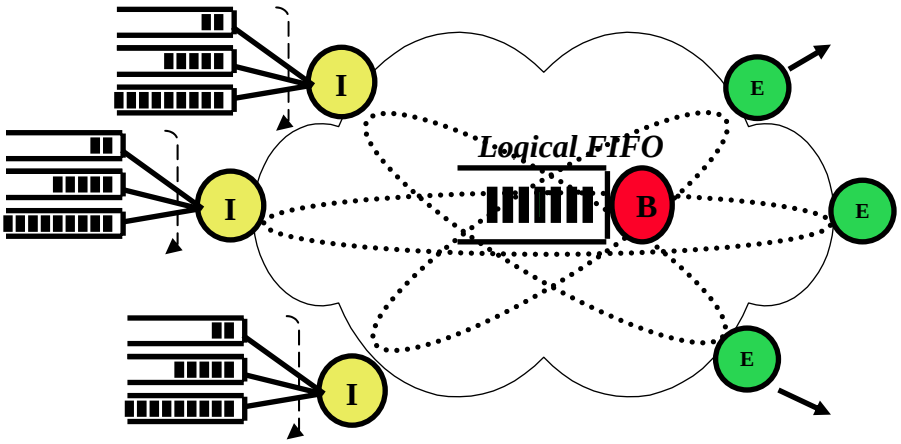

Figure 12. Overlay QoS Using Closed-Loop Control.

In summary, prior to this project, the problem of Quality of Service (QoS) was unsolved for multi-domain IP-based inter-networks primarily due to complex deployment and coordination issues. By "QoS" we refer to stable assurances on loss rate, bandwidth and delay. In contrast, QoS within a single administrative network today is largely a non-issue due to declining bandwidth costs and availability of administrative controls.

The Overlay QoS project focused on enabling QoS *on top of existing FIFO inter-networks*. This project has reduced the costs of engineering, provisioning and operating inter-networks with a range of QoS properties. The result of this research has been to enable a new paradigm, called "*QoS-on-a-dime*". The concept is that DoD can patch together an overlay network on top of heterogeneous allies' networks, and provision QoS on top of it using our fully distributed, light-weight tools! This has the potential to give the DoD dramatic tactical superiority and agility in a rapidly changing geo-political world and in a network-centric mode of warfare. We have provided a large-scale real-world demo of this capability (on the Planetlab infrastructure) in the DARPA NMS PI meetings.

To support the concept of Overlay QoS, we proposed a model to use accumulation, buffered packets of a flow inside network routers, to measure and control network congestion. This model is called accumulation-based congestion control (ACC). We developed a general control algorithm that includes a set of control policies. We proved its proportional fairness and global stability. ACC's router buffer scales to the number of sharing flows. We leveraged an adaptive virtual delay queuing (AVD) algorithm based upon work done at University of Illinois, Urbane Champaign which is incrementally deployable, to keep a small queue. This represents a major integration of a large class of congestion control methods into one theoretical framework.

The ACC model is then used as a closed-loop data-plane building block to provide weighted sharing and expected minimum rate services. Specifically we provide weighted service by allocating different accumulation among flows; we provide expected rate service by adaptively allocating accumulation to a flow such that the expected rate is achieved ***without the need for explicit admission control***. The relaxing of the admission control requirement, allowing graceful service degradation for lower priority flows is a dramatic advance over current technologies that may become unstable and blow up during such regimes of operation. We provide a steady state queuing and optimization analysis to show the theoretic results for a general network topology.

We implemented the Monaco scheme in Linux v2.2.18 using MIT Click software router as well as the ns-2 simulation package. We validated the theoretic results by extensive simulations and linux-based emulation experiments. The framework has been tested on the world-wide infrastructure called Planetlab (www.planet-lab.org, see Figure 11).

Figure 13. The World-wide Planetlab Infrastructure on Which Overlay QoS NMS Work Was Demonstrated.

4. Results and Discussion

The need for scalable and efficient for network analysis and management increases with the rapidly growing complexity and dynamics of the Internet. This project addresses this need at many different aspects.

In the area of a network experiment design, we have conducted scaling tests with Recursive Random Search (RRS) in which we handled thousands of parameter dimensions in difficult test optimization benchmarks. The RRS algorithm has been applied to the adaptive configuration of several real-world network protocols, such as RED, OSPF and BGP. The particular problems include RED parameter tuning, OSPF link weight tuning for traffic engineering, and BGP outbound load balancing using LOCAL_PREF tuning.

An extension of RRS, called a Unified Search Framework (USF), can be used to easily construct *on-the-fly*, a tailored optimization algorithm for a specific problem, that is run in a parallel way to fully utilize available computing resources. We have tested the USF on some benchmark functions and got very promising test results. The main advantage of the USF is its ability to function as a general platform to address various practical optimization problems with different features. It is especially suitable for those large-scale problems with simulation for function evaluations, such as the ones in on-line network parameter optimization.

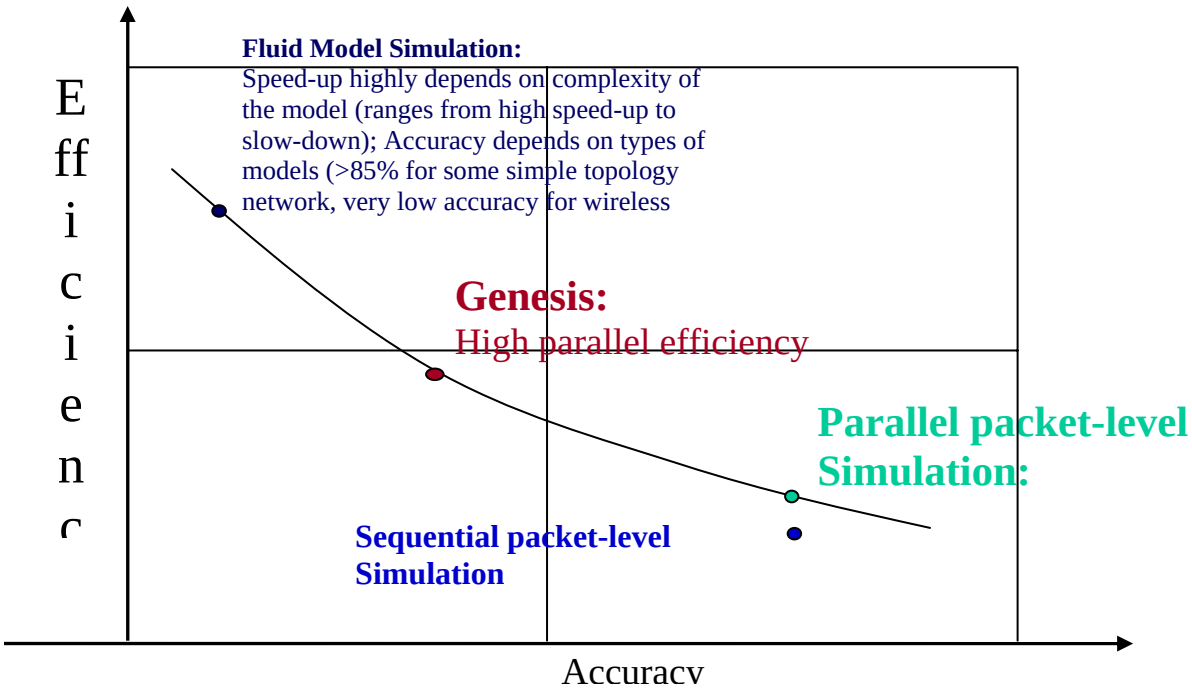

Figure 14. Tradeoff between Speed and Accuracy in Network Simulation Systems.

We have introduced a novel scheme, implemented in Genesis, to support scalable, efficient parallel network simulation. Our results indicate that the superlinear speedup for the single iteration step is possible and is the result of the non-linear complexity of the network simulation. Our approach achieved significant speedup in the simulations of different network scenarios. In addition to the speedup, the advantages of the presented method include fault tolerance, ability to integrate simulations and models in one run and support for truly distributed execution. When one of the participating processes fails, the rest can use the old delay and packet loss data to continue a simulation. When the only information available about a domain are delays across the domain and its outflows, the simulation of the other parts of the networks can directly use these data to perform the simulation.

We also demonstrated that our system can work efficiently with fully distributed network memory. This design reduces and makes scalable the memory size requirement for large-scale network simulations. As a result, Genesis is able to simulate huge networks using limited computer resources. The memory size required by BGP network simulation increases very quickly as the number of BGP routers and AS domains increases. Consequently, the simulation of large BGP networks was hindered by the memory size limitation. Genesis offers a new approach to simulating BGP on distributed memory that is scalable both in terms of simulation time and the required memory. In summary, Genesis offers a tradeoff between the simulation speed and accuracy, as shown in Figure 14.

By developing ROSS-Net, we have demonstrated that properly designed optimistic network simulators can be efficient and quite scalable. ROSS-Net was able to simulate million-node network topologies using inexpensive commercial off-the-shelf multiprocessor systems of a few dozen of processors consuming less than 1.4 GB of RAM in total. We have recently simulated the AT&T and several other large networks with full OSPF and TCP dynamics in a large-scale simulation platform. This configuration has tens of thousands of routers and over a million TCP flows interacting dynamically. This example suggests that we can now begin to simulate and make exploratory analysis of large-scale protocol feature interactions with our simulation and experiment design platform. Such complex feature interactions cannot be simulated, visualized or explored in small-scale ns-2 simulations.

In our theoretical work, we showed that timeouts and exponential backoffs are two main causes of scaling a TCP traffic and the degree of scaling is related to the loss rates seen by the TCP flows and the congestion on the links. These observations were then used to develop techniques to reduce the degree of scaling a TCP traffic including TCP pacing and active queue management strategies. We have also extended the packet interarrival model to obtain the MAC layer packet service times and consequently the distribution function of the delay in 802.11 based networks.

We developed analytic models to characterize the probability of the BGP convergence times exceeding certain bounds based on the network properties that led us to develop techniques to reduce the convergence time of a network of BGP routers. Our results show considerable reduction in the convergence times when our proposed modifications are incorporated into BGP.

In support of Overlay QoS, we developed a model called accumulation-based congestion control (ACC) to measure and control network congestion as well as a general control algorithm that includes a set of control policies. We proved its proportional fairness, global stability and its ability to scale with the number of sharing flows. We used this model to provide weighted sharing and expected minimum rate services. The relaxing of the admission control requirement, allowing graceful service degradation for lower priority flows is a dramatic advance over current technologies that may become unstable and blow up during such regimes of operation. We provide a steady state queuing and optimization analysis to show the theoretic results for a general network topology. We also validated the theoretic results by extensive simulations and linux-based emulation experiments. The framework has been tested on the world-wide infrastructure called Planetlab (www.planet-lab.org)

As part of the DARPA program we have had active collaboration with University of Illinois, Urbana Champaign, Caltech, University of California, Berkeley, Dartmouth College, Cisco and Lucent Technologies.

In addition, we have worked with SPAWAR whose representative, Mr. Phoung Nguyen visited RPI multiple times to study and access our technology. They have considered it as an option for their larger-scale integration project.

Also, we have had several visits from our AFRL monitors, Mr. John Valente and Dr. Kevin Kwiat, who have actively participated in RPI's research and have prioritized our technologies for transfer to AFRL. Dr. Kevin Kwiat was also a member of a dissertation committee for Dr. Yu Liu whose PhD thesis was based on this project.

These technology transfers occurred from 2002 to 2004.

The codes for Genesis is available at http://www.cs.rpi.edu/~liuy6/genesis
Other codes described in this report are available at
http://networks.ecse.rpi.edu/source_code.html

No patents were filled in connection with this project.

5. Conclusions

We have demonstrated that the network experiment design can be significantly improved with proper search frameworks. Our Recursive Random Search has been successfully applied to the adaptive configuration of several real-world network protocols, such as RED, OSPF and BGP. Its extension, Unified Search Framework is especially suitable for large-scale problems with simulation for function evaluations, such as the ones in on-line network parameter optimization. In future work, we will apply the USF to a variety of large-scale network optimization problems and demonstrate the flexibility of the USF in handling different problems.

Genesis constructively proved that scalable, efficient parallel network simulation achieving superlinear speedup at the cost of small decrease in accuracy of the simulation is possible. Our approach achieved significant speedup in the simulations of different network scenarios. The advantages of the approach include also fault tolerance, ability to integrate simulations and models in one run and support for truly distributed execution.

The ROSS-Net simulator developed in this project clearly demonstrates that optimistic parallel network simulators, when carefully designed, can provide high performance and scalability needed for on-line network management and optimization.

Our theoretical work shows that the analytical models of network traffic and protocols enable improvements in the desirable aspects of network performance. The techniques developed by us in the program from such analytical models include reduction of the TCP scaling and decrease of the time to convergence by BGP routers.

Overlay QoS based on accumulation-based congestion control (ACC) supports proportional fairness, global stability and ability to scale with the number of sharing flows. Using this model we were able to provide weighted sharing and expected minimum rate services and to relax the admission control requirement. This allows graceful service degradation for lower priority flows, creating a dramatic advance over current technologies that may become unstable and blow up during such regimes of operation.

Publications Related to the Project

1. D. Bauer, G. Yaun, C. D. Carothers, S. Kalyanaraman, and M. Yuksel, *Seven-O'Clock: A New Distributed GVT Algorithm Using Network Atomic Operations,* Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation (PADS '05), June 2005, to appear.

2. D. Bauer, G. Yaun, C. D. Carothers, M. Yuksel and S. Kalyanaraman, *Large-Scale Network Protocol Meta-Simulation Design and Performance Analysis,* Proceedings of the 2004 Winter Simulation Conference (WSC '04), December 2004.

3. K. Chandrayana, S. Ramakrishnan, B. Sikdar, S. Kalyanaraman, *On randomizing the sending times in TCP and other window based algorithms*, Computer Networks, to appear.

4. C. D. Carothers and B. K. Szymanski, *Checkpointing Multithreaded Programs,* Dr. Dobbs Journal, # 339, pages 46-51, August, 2002.

5. S. Deshpande and B. Sikdar, *On the Impact of Route Processing and MRAI Timers on BGP Convergence Times*, Proceedings of IEEE GLOBECOM, Dallas, TX, November 2004.

6. D. Harrison, Y. Xia, S. Kalyanaraman and A. Venkatesan, *An accumulation-based, closed-loop scheme for expected minimum rate and weighted rate services,* Computer Networks, Elsevier Science, Volume 45, Issue 6, pages 801-818, August 2004.

7. H.T. Kaur, T. Ye, and S. Kalyanaraman, *Minimizing Packet Loss by Optimizing OSPF Weights using On-line Simulation,* Proceedings of 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003), Pages 79-86, Orlando, Florida, October 2003.

8. H.T. Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi, *BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet,* Proceedings of ACM SIGCOMM Workshop on Future Directions on Network Architectures (FDNA), Volume 33, Issue 4, Pages 277-288, Karlsruhe, Germany, August 2003.

9. J. Levy-Vehel and B. Sikdar, *A Multiplicative Multifractal Model for TCP Traffic,* Proceedings of IEEE ISCC, pp. 714-719, Hammamet, Tunisia, July 2001.

10. Y. Liu, and B. Szymanski, *Distributed Packet-Level Simulation for BGP Networks under Genesis,* Proceedings of 2004 Summer Computer Simulation Conference (SCSC'04), July 25 - 29, 2004, San Jose, CA.

11. Y. Liu, B.K. Szymanski and A. Saifee, *Genesis: A Scalable Distributed System for Large-scale Parallel Network Simulation,* Computer Networks Journal, to appear 2005.

12. K. Mandani, and B. Szymanski, *Integrating Distributed Wireless Simulation Into Genesis Framework,* Proceedings of the Summer Computer Simulation Conference, July 2003.

13. B. Sikdar, K. Chandrayana, K. S. Vastola and S. Kalyanaraman, *Queue management algorithms and network traffic self-similarity,* Proceedings of IEEE HPSR, pp. 319-323, Kobe, Japan, May 2002.

14. B. Sikdar, K. Chandrayana, K. S. Vastola and S. Kalyanaraman, *On Reducing the Degree of Second-Order Scaling in Network Traffic,* Proceedings of IEEE GLOBECOM, pp. 2594-2598, Taipei, Taiwan, November, 2002.

15. B. Sikdar, S. Kalyanaraman and K. S. Vastola, *Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK,* IEEE/ACM Transactions on Networking, vol. 11, no. 6, pp. 959-971, December 2003.

16. B. Sikdar and K. S. Vastola, *The Effect of TCP on the Self-Similarity of Network Traffic,* Proceedings of 35th Conference on Information Sciences and Systems, Baltimore, MD, March 2001.

17. B. Sikdar and K. S. Vastola, *On the Contribution of TCP to the Self-Similarity of Network Traffic,* Lecture Notes in computer Science (Proceedings of IWDC), vol. 2170, pp. 596-613, 2001.

18. B.K. Szymanski, Q. Gu, and Y. Liu, *Time-Network Partitioning for Large-Scale Parallel Network Simulation Under SSFNet,* Proceedings of the Applied Telecommunication Symposium, ATS2002, B. Bodnar (edt), San Diego, CA, April 14-17, SCS Press, pp. 90-95.

19. B.K. Szymanski, Y. Liu and R. Gupta, *Parallel Network Simulation under Distributed Genesis,* Proceedings of the 17th Workshop on Parallel and Distributed Simulation, June 2003.

20. B.K. Szymanski, Y. Liu, A. Sastry, and K. Madnani, *Real-Time On-Line Network Simulation,* Proceedings of the 5th IEEE International Workshop on Distributed Simulation and Real-Time Applications DS-RT 2001, August 13-15, 2001, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 22-29.

21. B.K. Szymanski, A. Saifee, A. Sastry, Y. Liu and K. Madnani, *Genesis: A System for Large-scale Parallel Network Simulation,* Proceedings of the 16th Workshop on Parallel and Distributed Simulation, pp. 89--96, May 2002.

22. B.K. Szymanski, J.-F. Zhang, and J. Jiang, *A Distributed Simulator for Large-Scale Networks with On-Line Collaborative Simulators,* Proceedings of the European Multi-simulation Conference - ESM99, Warsaw, Poland, SCS Press, II:146--150, June, 1999.

23. O. Tickoo and B. Sikdar, *Modeling and Analysis of Traffic Characteristics in IEEE 802.11 MAC Based Networks,* Proceedings of IEEE GLOBECOM, pp. 67-71, Taipei, Taiwan, November, 2002.

24. O. Tickoo and B. Sikdar, *On the impact of IEEE 802.11 MAC on traffic characteristics,* IEEE Journal on Selected Areas in Communications, vol. 21, no. 2, pp. 189-203, February 2003.

25. O. Tickoo and B. Sikdar, *Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC based Wireless Networks,* Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.

26. O. Tickoo and B. Sikdar, *A Queueing Model for Finite Load IEEE 802.11 Random Access MAC,* Proceedings of IEEE ICC, Paris, France, June 2004.

27. Y. Xia, D. Harrison, S. Kalyanaraman, K. Ramachandran and A. Venkatesan, *An Accumulation-based Congestion Control Model,* Proceedings of IEEE International Conference on Communications (ICC), Volume 1, Pages 657-663, Anchorage, Alaska, May 2003.

28. Y. Xia, D. Harrison, S. Kalyanaraman, K. Ramachandran, A. Venkatesan, *Accumulation-based Congestion Control,* IEEE/ACM Transactions on Networking, 2005, to appear.

29. G. Yaun, H. L. Bhutada, C. D. Carothers, M. Yuksel, and S. Kalyanaraman, *Large-Scale Network Simulation Techniques: Examples of TCP and OSPF Models,* ACM SIGCOMM Computer Communication Review Special Issue on Tools and Technologies for Networking Research and Education, Volume 33, No 3, pp. 27-41, July, 2003.

30. G. Yaun, C. D. Carothers, and S. Kalyanaraman, *Large-Scale TCP Models Using Optimistic Parallel Simulation,* Proceedings of the 17th Workshop on Parallel and Distributed Simulation (PADS '03), June 2003. **Best paper award.**

31. T. Ye, D. Harrison, B. Mo, B. Sikdar, H. T. Kaur, S. Kalyanaraman, B. Szymanski, and K. Vastola, *Network Management and Control using Collaborative On-Line*

Simulation, Proceedings of IEEE International Communications Conference (ICC), Volume 1, Pages 204-209, Helsinki, Finland, June 2001.

32. T. Ye, and S. Kalyanaraman, *Adaptive Tuning of RED Using On-line Simulation,* Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), Volume 3, Pages 2210-2214, Taipei, Taiwan, November 2002.

33. T. Ye, and S. Kalyanaraman, *A Recursive Random Search Algorithm for Large-Scale Network Parameter Configuration*, Proceedings of ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, Pages 196-205, San Diego, California, June 2003.

34. T. Ye, and S. Kalyanaraman, *A Recursive Random Search Algorithm for Black-box Optimization,* ACM Performance Evaluation Review Journal, December 2004.

35. T. Ye, H.T. Kaur, and S. Kalyanaraman, *Large-Scale Network Parameter Configuration Using An On-line Simulation Framework,* IEEE/ACM Transactions of Networking, under review.

36. M. Yuksel, B. Sikdar, K. S. Vastola, and B. Szymanski, *Workload Generation for ns Simulations of Wide Area Networks and the Internet,* Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, 93--98, SCS Press, 2000.

Ph.D. Theses Supported By the Project

1. Biplab Sikdar, August 2001
2. David Harrison, December, 2001
3. Hema Tahilramani, December 2002
4. Ye Tao, April 2003
5. Yong Xia, Spring 2004 (awarded the Charles Close Dissertation Prize)
6. Yu Liu, September 2004